社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－CSEC－37 (8)
2007／5／25

隣接行列を用いたアクセス制御ポリシーの統合法

河野和宏[†] 伊藤義道[†] 青山明史[††]
鴨田浩明[†,††] 馬場口登[†]

近年，組織内部からの不正行為や単純なミスによる情報漏洩が問題となっている．これらを防ぐため，使用するアプリケーションやシステムに対し，アクセス制御ポリシーを設定することで対処がなされている．しかしながら，各システムのアクセス制御ポリシーはそれぞれのシステムで独自に記述されているため，組織におけるセキュリティガイドラインが変更されたり，新しいシステムが導入された場合には，管理者は全てのシステムに対し個々にポリシーを設定しなおす必要がある．そこで本稿では，対象となるポリシーの集合から，それらを一括して管理することが可能な統合されたポリシーを作成する手法を提案する．提案方式では，グラフ理論を用いて統合を行っており，アルゴリズムを容易に構築することができる．また，グラフの表現として隣接行列を用いることで，グラフの変形や構築を全て行列演算に帰着している．

An integration method of access control policies using adjacency matrix

Kazuhiro Kono,[†] Yoshimichi Ito,[†] Akihito Aoyama,[††]
Hiroaki Kamoda[†,††] and Noboru Babaguchi[†]

Recently, it becomes a serious problem that information leakage occurs by fraudulent action from the inside and by mis-operation. To prevent such a problem, organizations introduce access control management systems and set access control policies to application softwares and systems. However, the access control policy in each system is defined independently, administrators have to set the polices for all the systems when their security guideline is changed and new systems are introduced. In this paper, we propose a method for generating an integrated policy so as to manage every policy in each system and application at the same time. In the proposed method, we integrate these polices using graphs and corresponding adjacency matrices. By using them, it is shown that the problem of integrating policies can be reduced to that of matrix operations, and this enables us to constitute the integration algorithm easily.

1. はじめに

近年，ユーザの不正行為や単純なミスによる内部からの情報漏洩が深刻な問題となっている．この対策として，アクセス制御システムを導入し，ユーザに対してアクセス制限をかけることで対応がなされている．

現代の情報システム社会では，アクセス制御システムは様々な状況で活用されている．具体的には，ネットワーク，およびサービスへの制御システム，共有ディレクトリや共有ファイルへのアクセス制御，さらに，ファイルへのより詳細な制御が可能となるファイル保護システムが存在する．例えば，ファイル保護システムとして，Microsoft 社の office 文書を制御する Microsft Windows®Rights Management Services(RMS)[1)]，Adobe 社の pdf 文書を制御する Adobe®Livecycle™Policy Server[2)] が挙げられる．これらはともに，ファイル単位できめ細かなアクセス権を設定する機能を提供し，システム管理者はこれらのシステムに対応したアクセス制御ポリシーを設定する．

一般に，アクセス制御システムのポリシーは，それぞれのシステムで独自に記述されている．そのため，各システムはそれぞれ似たようなポリシーを設定しているにもかかわらず，管理者は全てのシステムに対して同じような手間をかけて個々にポリシーを設定する必要がある．例えば，個人情報保護法や社内のセキュリティガイドラインが変更された場合，管理者はそれぞれ個々に対応する必要があり，多大な時間と労力が必要となる．これは，ポリシーを一括して管理する

† 大阪大学大学院工学研究科
†† 株式会社 NTT データ

枠組みが存在しないためであり，今後のアクセス制御システムの複雑化，および分散化を考慮すると，ポリシーの統合技術は必要不可欠であると考えられる．

ポリシーの統合に関する先行研究として，文献 3), 4) が挙げられる．文献 3) では，異なる組織間で共有するデータにアクセスする方法を提案しており，相互のモデルにおいて，一方を基本モデルとし，他方をそのモデルに合わせる方法を提案している．文献 4) では，異なる環境にあるポリシーに矛盾が存在しないように，整数計画法を用いてモデルの調停を行い，ポリシーの統合を行っている．

これらの研究は，ともに RBAC (Role-Based Access Control) におけるポリシーの統合を対象にしている．RBAC とは，ロール (サブジェクトを組織的・機能的な役割に基づいて振り分けたグループ) に基づいてサブジェクト，リソース，アクションのアクセス制御を行う方式であり，下位のロールが有するアクセス権は全て上位のロールに継承されるという継承関係が存在する (例えば，課長の有するアクセス権は部長も持つ等)．

一方，このような継承関係は，一般にはリソースやアクションに対しても存在する．例えば，ファイル保護システムである Widows RMS や Adobe Livecycle Policy Server では，アクションに関して継承関係が存在している (編集できる人は印刷もできる等)．したがって，サブジェクト，リソース，アクションのそれぞれのカテゴリに対して継承関係を考慮したモデルを考察する方がより現実的であるといえる．本稿では，これら全てのカテゴリに対して継承関係を考慮したモデルを導入し，対象となるポリシーの集合からそれらを一括して管理することが可能な統合されたポリシーを作成する手法を提案する．

また，先行研究は，主に複雑な論理式を用いて議論が展開されるため，必ずしも見通しの良い統合手法を与えるものとはなっていない．本稿で提案する手法は，継承関係を表現する際に論理式ではなくグラフを用いているため，より直観的に把握し易い統合手法を与えるものとなっている．本稿ではさらに，グラフの表現として隣接行列を用いることで，統合の際に必要な計算を全て行列の演算に帰着できることを示す．これにより，見通し良くポリシーを統合するアルゴリズムを実装することが可能となる．

2. アクセス制御モデル

本稿では，アクセス制御を，『あるサブジェクトが，どのリソースに対して，いかなるアクションができるかを許可する』機能と定義し，これらを記述したものをアクセス制御ポリシー (以下，ポリシー) と呼ぶ．また，本稿では，サブジェクト，アクション，リソースのそれぞれのカテゴリに対して継承関係が存在するモデルを対象とする．各カテゴリにおける継承関係の例を，図 1〜図 3 に示す．

例えば，図 1 において，“編集” から “表示” に向かう矢印は，『あるサブジェクトが，あるリソースに対して，“編集” できるならば，そのサブジェクトは，そのリソースに対して，“表示” できる』という継承関係を意味する．図 2 においては，『“一般社員” が，あるリソースに対して，あるアクションを実行できるならば，“役員” は，そのリソースに対して，そのアクションを実行できる』という意味である．同様に図 3 では，『あるサブジェクトが，“秘密文書” に対して，あるアクションを実行できるならば，そのサブジェクトは，“公開文書” に対して，そのアクションを実行できる』という意味をもつ．

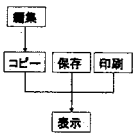

図 1 アクションの継承関係の例

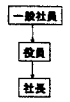

図 2 サブジェクトの継承関係の例

図 3 リソースの継承関係の例

一般に継承関係は，『サブジェクト s_i がリソース r_k に対してアクション a_m が許可されるならば，サブジェクト s_j がリソース r_l に対してアクション a_n が許可される』と記述できる．これを，$(s_i, r_k, a_m) \rightarrow (s_j, r_l, a_n)$ と略記する．この記法を用いると，図 1 における”編集”と”コピー”の継承関係は，

$$(s_i, r_k, \text{編集}) \rightarrow (s_i, r_k, \text{コピー})$$
$$(\forall s_i \in SUB, \forall r_k \in RES)$$

と表される．ただし，SUB, RES は，それぞれサブジェクトの集合，およびリソースの集合である．

この例におけるアクションの継承関係は，任意のサブジェクトやリソースに対し無関係に成立するものである．このように，ある継承関係が他のカテゴリにおける継承関係とは無関係に成立することを独立と呼ぶ．本稿では，他のカテゴリとは独立な継承関係のみを扱う．したがって，上の例のような場合は”編集”→”コピー”と略記することにする．

3. 統合ポリシーの作成，および継承関係の統合の概要

本節では，統合ポリシー作成までの枠組み，および継承関係の統合について説明する．

3.1 統合ポリシー作成の方針

本稿で扱うモデルでは，サブジェクト，リソース，アクションの各カテゴリには継承関係が存在する．また，各カテゴリの継承関係は，一般にアプリケーションやシステムごとに異なっている．

本稿では，ポリシーを作成するために，次の3段階に分けて作成する手法を提案する．

(1) 各カテゴリごとに継承関係を統合する．
(2) 統合された3つのカテゴリの継承関係を，1つの継承関係に統合する．
(3) 管理者が基本的なポリシーを設定する．その後，統合した継承関係を参照することで，他のポリシーも自動的に設定される．

3.2 グラフに関する基本事項

図1〜図3に示したとおり，継承関係は有向グラフで表現される．本節では，継承関係を表現する際に必要となるグラフ理論について説明する．

グラフとは，頂点と呼ばれる要素の集合 V と，V の相異なる2つの頂点の順序対を要素とする枝と呼ばれる要素の集合 B から構成される集合の対 $G(V,B)$ のことである．頂点の集合 V と枝の集合 B をとくに明記する必要がない場合，グラフを単に G と表記する．

本稿で扱うグラフは，すべての枝に向きが決められている有向グラフである．また，本稿では，相異なる2つの頂点を結ぶ2本以上の枝や，同一の頂点を結ぶ枝を持たない単純グラフを扱う．

次に，グラフ $G(V,B)$ において，歩道とは，$G(V,B)$ の頂点と枝とが交互に現れる有限列 $v_1b_1v_2b_2\cdots b_{n-1}v_n$ であり各枝 b_i が v_i と v_{i+1} を端点としてもつものである．v_1 を歩道の始点，v_n を終点という．$b_i \neq b_j$ $(i \neq j)$，かつ $v_i \neq v_j$ $(i \neq j)$ である歩道を道という．また，ある歩道において，$v_1 = v_n$，すなわち，始点と終点が一致しているとき，その歩道は閉じていると定義し，閉じている道を閉路と呼ぶ．

統合の対象となるアクセス制御システム ACS_k のアクションの集合を ACT_k，それらのアクションに対する個々の継承関係の集合を $ACTR_k$ とすると，ACS_k のアクションの継承関係全体はグラフ $G_k(ACT_k, ACTR_k)$ で表現できる．以下，同様に，ACS_k のサブジェクトの継承関係全体，およびリソースの継承関係全体を，それぞれ $G_k(SUB_k, SUBR_k)$，$G_k(RES_k, RESR_k)$ で表す．

3.3 同一カテゴリにおける継承関係の統合

継承関係の統合は，同一カテゴリにおける統合をそれぞれのカテゴリに対して行なった後，3つのカテゴリの継承関係を統合することで達成される．以下では，同一カテゴリ (つまり，アクションならアクション同士) における統合について述べる．

3.3.1 同一カテゴリにおける継承関係の基本操作

統合の対象となる N 個のシステムにおける同一カテゴリの継承関係を，それぞれ $G_k(ACT_k, ACTR_k)$ とする (ただし，k は $1 \leq k \leq N$)．このとき，同一カテゴリにおける継承関係の統合の基本操作は，対象となる継承関係の和集合をとる操作である．式で表すと以下のとおりである．ただし，継承関係の和集合をとった後の，継承関係を $G(ACT, ACTR)$ とし，$a_k(k = 1, \cdots, n_A)$ はアクション，n_A はアクションの数を表す．

$$ACT = ACT_1 \cup ACT_2 \cup \cdots \cup ACT_N = \{a_1, a_2, \cdots, a_{n_A}\} \quad (1)$$

$$ACTR = ACTR_1 \cup ACTR_2 \cup \cdots \cup ACTR_N \quad (2)$$

しかし，和集合をとる操作だけでは，閉路や冗長な枝に起因する問題が生じる．次の小節から閉路，および冗長な枝に関する問題について述べる．

3.3.2 閉　　路

閉路とは3.2節で記述したとおり，グラフの頂点を結ぶ閉じた道のことである．閉路の例を図4に示す．

統合した結果，閉路が存在した場合，閉路を構成する頂点は同等な関係となるため，扱いが困難になる．そこで，提案手法では，閉路の頂点を全て1つの頂点として再定義し，閉路を削除する方策をとる．閉路を検出し，削除する方法については4.2節で述べる．

3.3.3 冗 長 性

冗長性は枝に関する性質であり，次のように定義される．

定義1 閉路が存在しない有向グラフ G の頂点の集合を $V = \{v_1, v_2, \cdots, v_p\}$ とし，ある2つの頂点 v_i から v_j (ただし，$i \neq j$ で，$1 \leq i \leq p$ および $1 \leq j \leq p$) へと向かう枝 b_{ij} があるとする．このとき，v_i から v_j へと至る長さ2以上の道が存在する場合，その枝 b_{ij} を冗長な枝と定義する．

冗長な枝の例を図5に示す．図5において冗長な枝は破線の枝である．

v_i から v_j へ向かう枝が冗長であるとき，その枝がなくても，推移律から v_i と v_j の関係がわかる．したがって，その枝は余分な情報となる．冗長な枝の数が増加すると，それだけ余分な情報の増加につながる．そこで，提案手法では，冗長な枝を全て削除する．冗長な枝を検出し，削除する方法については4.2節で述べる．

3.3.1節，3.3.2節，および3.3.3節から，同一カテゴリにおける継承関係の統合とは，以下の3つの操作

で実施される．

(1) 対象となる複数の継承関係の和集合をとる．

(2) 和集合をとった継承関係に対して，閉路を検出し，グループ化を行い削除する．

(3) 閉路が存在しない継承関係に対して，冗長な枝を検出し，削除する．

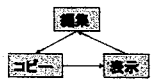

図 4 閉路

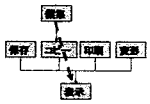

図 5 冗長な枝

4. 隣接行列を用いた統合ポリシーの作成手法

前節では，同一カテゴリにおける継承関係の統合について説明し，必要な操作を記述した．本節では，隣接行列を用いて同一カテゴリの統合におけるこれらの操作を行う手法を述べる．また，異なるカテゴリにおける継承関係の統合についても説明し，本節の最後で統合ポリシーの作成までのプロセスを示す．

4.1 隣接行列とその性質

グラフの表現，およびその性質を調べる手法の一つとして，隣接行列を用いる方法がある．以下，本節では隣接行列，およびその加法の定義と，基本的な性質を述べる．

定義 2 $V=\{v_1,v_2,\cdots,v_n\}$ を頂点として持つ単純な有向グラフの隣接行列 $R=[r_{ij}]$ を次のように定義する．ただし，i,j は $1\leq i\leq n$，$1\leq j\leq n$ である．

$$r_{ij}=\begin{cases}1 & (v_i\text{から}\ v_j\text{へ向かう枝が存在する})\\ 0 & (\text{それ以外})\end{cases} \tag{3}$$

定義 3 隣接行列の加法は，頂点集合が等しいグラフの隣接行列に対して定義される．頂点集合の等しい 2 つのグラフの隣接行列 $A=[a_{ij}]$，および $B=[b_{ij}]$ に対して，加法 $A+B$ は要素ごとの和で定義する．ただし，各要素における和は，論理和である．すなわち，

$$a_{ij}+b_{ij}=\begin{cases}0 & (a_{ij}=b_{ij}=0)\\ 1 & (\text{それ以外})\end{cases} \tag{4}$$

とする．

このとき，v_i から v_j へ向かう長さ p の道が存在するための条件として，以下の結果が知られている．

補題 1 隣接行列 R で表現される有向グラフ G の頂点集合を $V=\{v_1,v_2,\cdots,v_n\}$ とする．このとき，v_i から v_j へ向かう長さ p の道が存在するための必要十分条件は，R^p の (i,j) 成分が 1 となることである．

4.2 隣接行列を用いた統合ポリシーの作成手法

本小節では，隣接行列を用いた統合ポリシーの作成手法について 2 段階に分けて述べる．

4.2.1 同一カテゴリにおける継承関係の統合手法

まず，サブジェクト，リソース，アクションの各カテゴリにおける継承関係を統合する手法について説明する．各々同じ手法を用いて統合するので，本節ではアクションの継承関係を統合する手法について述べる．なお，継承関係を隣接行列で表現する際，行列の加算および乗算を行うため，統合の対象となる各システムのアクションの集合は一致する必要がある．このため，N 個のシステムのアクションの継承関係を表すグラフは，あらかじめ $ACT=\{a_1,a_2,\cdots,a_{n_A}\}$ を頂点の集合としたもので記述しておく必要がある．また，隣接行列で表現する際は，アクションの並び方が全てのシステムで同じ順序になっている必要がある．

継承関係の和集合をとる操作

基本操作である継承関係の和集合をとる操作は，対象となる隣接行列の加算を実行することに相当する．それぞれのシステムにおいて，アクションの継承関係を表す隣接行列を A_k，統合されたアクションの継承関係を表す隣接行列を A とすると，これらは次の式によって関係づけられる．

$$A=A_1+A_2+\cdots+A_N \tag{5}$$

継承関係の閉路の検出/削除

閉路の存在に関して，補題 1 から以下の定理が導かれる．

定理 1 隣接行列 R で表現される有向グラフ G は n 個の頂点を持つとする．このとき，グラフ G に閉路が存在しないための必要十分条件は，$R^n=0$ となることである．

定理 2 隣接行列 R で表現される有向グラフ G において，v_i を含む長さ l の閉路が存在するための必要十分条件は，R^l の (i,i) 成分が 1 となることである．

定理 1 の証明は，本稿では省略する．定理 1，および定理 2 から，継承関係の閉路の検出，および削除方法は次の手順で実行される．ただし，閉路の削除方法に関する証明は省略する．

(1) $l=2$ とする．

(2) A の l 乗を計算する．

(3) A^l の (i,i) 成分が 1 である場合，次の i)〜vi) を行う．

(i) 閉路となる l 個の頂点をそれぞれ，$\{c_1,c_2,\cdots,c_l\}$ とし，このうち，ある頂点 c_m に対して以下の操作を実行する．

(ii) 閉路となる全ての頂点の列の論理和を計算し，計算結果を第 m 列と置き換える．

(iii) 閉路となる全ての頂点の行の論理和を計算し，計算結果を第 m 行と置き換える．

(iv) (m,m) 成分を 0 にする.

(v) 第 m 行と第 m 列を除く，閉路となる頂点を持つ行，および列を行列から除き，新たにできる行列を A と再定義する.

(vi) l を 2 にして，(2) に戻る.

(4) A^l のすべての成分が 0 ならば，グラフには閉路が存在しないので，このアルゴリズムを終了する.

(5) l を 1 増やして (2) に戻る.

冗長な枝の検出/削除

冗長な枝の存在に関しては，補題 1 から次の定理 3 が導かれる.

定理 3 隣接行列 R で表現される，閉路が存在しない有限グラフ G について考える．このとき，要素 v_i から要素 v_j へ向かう枝が冗長であるための必要十分条件は，隣接行列 R に対して，ある $l \geq 2$ が存在し，R^l の (i,j) 成分が 1 となることである.

定理 3 と定理 1 から，冗長な枝の検出，および削除の手法は次の手順で実行される.

(1) $l = 2$ とする.

(2) A の l 乗を計算する.N は A の頂点の数である.

(3) A，および A^l の (i,j) 成分がともに 1 である場合，v_i から v_j へ向かう枝は冗長であるとして，(i,j) を記憶する.

(4) A^l の全ての成分が 0 ならば，冗長な枝として記憶した A の成分を全て 0 にし，このアルゴリズムを終了する.

(5) l を 1 増やして (2) に戻る.

以上に述べた 2 つのアルゴリズムは，定理 1 によって有限回の操作で終了することが保証される.

4.2.2 異なるカテゴリにおける継承関係の統合手法

本小節では，サブジェクト，リソース，アクションの異なる継承関係を統合する手法について説明する．継承関係の統合は，グラフを描いて直観的に行なうこともできるが，枝の与え方によって様々なバリエーションが存在するため，扱いが非常に厄介である．また，この方法ではアルゴリズム化して実装することも困難である．そこで本稿では，Policy Product という演算を定義し，その演算に基づいて継承関係を統合する．Policy Product とは，2 つの異なるカテゴリの継承関係を統合する行列の演算であり，以下のように定義する.

定義 4 A, B を，互いに異なるカテゴリの継承関係を表す隣接行列とする．ただし，A は n 次の正方行列，B は m 次の正方行列とする．このとき，A と B の Policy Product を $A \otimes_p B$ と表記し，次式で定義する.

$$A \otimes_p B := A \otimes I_m + I_n \otimes B$$

ただし，$\otimes$ はクロネッカー積であり，I_m は m 次の単位行列を表す．なお，k 行 l 列の行列 $X = [x_{ij}]$ と q 行 r 列の行列 Y のクロネッカー積 $X \otimes Y$ は kq 行 lr 列の行列であり，(6) 式で定義される.

$$X \otimes Y := \begin{bmatrix} x_{11}Y & \cdots & x_{1l}Y \\ \vdots & & \vdots \\ x_{k1}Y & \cdots & x_{kl}Y \end{bmatrix} \quad (6)$$

以上で定義した Policy Product を用いて，まずサブジェクトとアクションの継承関係を統合する手法を述べる.

サブジェクトの継承関係を $G(SUB, SUBR)$，アクションの継承関係を $G(ACT, ACTR)$ とする．また，それらを表す隣接行列を，それぞれ，S, A とする．このとき，サブジェクトとアクションの Policy Produt を行った行列 $S \otimes_p A$ は，$SUB \times ACT$ 上の継承関係を表す隣接行列となることが確認できる．ただし，$SUB \times ACT$ は，$SUB = \{s_1, \cdots, s_{n_S}\}$，$ACT = \{a_1, \cdots, a_{n_A}\}$ とすると，次のようになる.

$$SUB \times ACT =$$
$$\{s_1a_1, \cdots, s_1a_{n_A}, \cdots, s_{n_S}a_1, \cdots, s_{n_S}a_{n_A}\}$$

さらに，リソースの継承関係 $G(RES, RESR)$ を統合するには，$P = S \otimes_p A \otimes_p R$ を計算する．ただし，R はリソースの継承関係を表す行列である．$RES = \{r_1, \cdots, r_{n_R}\}$ とすると，P は次式で定義される集合 $SUB \times ACT \times RES$ 上の継承関係を表す.

$$SUB \times ACT \times RES =$$
$$\{s_1a_1r_1, \cdots, s_1a_1r_{n_R}, \cdots, s_1a_{n_A}r_1,$$
$$\cdots, s_1a_{n_A}r_{n_R}, \cdots, s_{n_S}a_1r_1, \cdots, s_{n_S}a_1r_{n_R},$$
$$\cdots, s_{n_S}a_{n_A}r_1, \cdots, s_{n_S}a_{n_A}r_{n_R}\}$$

以上により，異なる 3 つのカテゴリ全ての継承関係の統合が Policy Product により可能となる.

ここで，前節で述べたように，統合された継承関係が閉路や冗長な枝を含むかどうかを検討する必要がある．これに関しては，以下の定理が得られる.

定理 4 隣接行列 A，および B で表現される継承関係が，ともに閉路を含まないならば，隣接行列 $A \otimes_p B$ で表現される継承関係は閉路を含まない.

定理 5 隣接行列 A，および B で表現される継承関係が，ともに閉路と冗長な枝を含まないならば，隣接行列 $A \otimes_p B$ で表現される継承関係は冗長な枝を含まない.

定理 4，および定理 5 の証明は，補題 2 を用いる．本稿では，定理 4 のみ証明する.

補題 2 A は n 次の正方行列，B は m 次の正方行

列とする．このとき，以下の関係が成立する．

$$(I_m \otimes A)^k = I_m \otimes A^k, \quad (7)$$

$$(B \otimes I_n)^k = B^k \otimes I_n, \quad (8)$$

$$(A \otimes I_m) \cdot (I_n \otimes B) = (I_n \otimes B) \cdot (A \otimes I_m) \quad (9)$$

$$(A \otimes_p B)^k = \sum_{l=0}^{k} (A \otimes I_m)^l \cdot (I_n \otimes B)^{k-l} \quad (10)$$

定理 4 の証明 隣接行列 A, B は，それぞれ，n 次，および，m 次の正方行列であり，対応するグラフはともに閉路を持たないとする．このとき，定理 1 より，

$$A^l = 0 \ (\forall l \geq n), \quad B^k = 0 \ (\forall k \geq m) \quad (11)$$

となる．これらの条件より，$(A \otimes_p B)^{mn} = 0$ となることを示せば，$A \otimes_p B$ で表現されるグラフは閉路を含まないことが，定理 1 より示される．$\bar{A} = A \otimes I_m, \bar{B} = I_n \otimes B$ とおくと，補題 2 より以下の式を得る．

$$\bar{A}^l = 0 \ (\forall l \geq n). \quad (12)$$

$$\bar{B}^k = 0 \ (\forall k \geq m). \quad (13)$$

$$(A \otimes_p B)^{mn} = \sum_{l=0}^{mn} \bar{A}^l \bar{B}^{mn-l}. \quad (14)$$

また，$(m-1)(n-1) \geq 0$ より，$mn - n + 1 \geq m$ が導かれる．これらの式より，

$$(A \otimes_p B)^{mn} = \sum_{l=0}^{n-1} \bar{A}^l \bar{B}^{mn-l} \quad (15)$$

$$= \sum_{k=mn-n+1}^{mn} \bar{A}^{mn-k} \bar{B}^k = 0 \quad (16)$$

を得ることができる．(証明終)

以上の定理から，次の系が得られる．

系 1 隣接行列 A，および B で表現される継承関係が，ともに閉路と冗長な枝を含まないならば，隣接行列 $A \otimes_p B$ で表現される継承関係は閉路と冗長な枝を含まない．

したがって，統合前の継承関係が全てのカテゴリにおいて閉路と冗長な枝を含まなければ，Policy Product を用いて統合された継承関係は，あらたにチェックをするまでもなく，閉路と冗長な枝を含まないことが保証されることになる．

また，以上に述べた方法を用いて，サブジェクト，アクション，リソースの 3 つの継承関係を統合する際は，それらのカテゴリにおいて閉路と冗長性がない場合，$S \otimes_p A \otimes_p R$ の計算によって達成できる．

4.3 ポリシーを作成するまでのプロセス

本章の最後に，ポリシーを作成するまでのプロセスをまとめる．

ポリシーを作成するまでのプロセスは 4 段階あり，以下の通りである．

(1) 同じカテゴリの継承関係の統合

(2) 異なるカテゴリの継承関係の統合

(3) 管理者によるポリシーの設定

(4) XACML などのポリシー記述言語に出力

1 つ目，および 2 つ目のステップでは，継承関係の統合を行う．その後，この統合された継承関係を用いてポリシーの作成を 3 つ目のステップから行う．

3 つ目のステップでは，まず管理者がポリシーを設定する．その後，隣接行列を用いて，本来なら可能な操作に対してもポリシーを設定する．これにより，管理者の手間を省くとともに，可能な操作を出力することで，正しくポリシーを設定することが可能となる．

最後のステップでは，3 つ目のステップで出力されたポリシー全てを XACML[6] として出力する．

5. 結　　論

本稿では，サブジェクト，リソース，アクション全てに継承関係を導入したアクセス制御モデルを対象とし，複数のアクセス制御システムを，一括して管理可能なアクセス制御ポリシーを作成する手法を提案した．

今後の課題として，システム内のポリシーが正しく設定されているか検証する手法に関する問題や，実装の問題，さらにはグループ化による読み替えを各システムに反映させる手法などがある．特に実装に関しては，最終的に作成される行列が巨大なスパース行列となることから，効率的な行列の演算が必要となる．

参 考 文 献

1) Microsoft：Windows rights management services, http://www.microsoft.com/japan/windowsserver2003/technologies/rightsmgmt/default.mspx.

2) Adobe Systems：Adobe livecycle policy server, http://www.adobe.com/jp/products/server/policy/.

3) C. Pan, P. Mitra, P. Liu:Semantic Access Control for Information Interoperation, *Proc. of 11th ACM symposium on Access control models and technologies*, pp. 237-246 (2006).

4) B. Shafiq, James B. D. Joshi, E. Bertino, A. Ghafoor：Secure Inteoperation in a Multidomain Environment Employing RBAC Policies, *IEEE Trans. KNOWLEDGE AND DATA ENGINEERING*, vol. 17, No. 11, pp. 1557-1577, Nov (2005).

5) D. F. Ferraiolo, D. R. Kuhn：Role based access control, *15th National Computer Security Conference*, Available from http://csrc.nist.fov/rbac/ (1992).

6) XACML and OASIS Security Services Technical Committee：eXtensible Access Control Markup Language(xacml) committee specification 2.0, Feb (2005).